УДК 611.08

М.А. Гайдучок

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВІЗУАЛІЗАЦІЇ ФІЛЬТРАЦІЇ ТА СПЕКТРАЛЬНОГО АНАЛІЗУ БІОМЕТРИЧНИХ МЕРЕЖЕВИХ ДАНИХ

Вінницький національний технічний університет

Анотація

Визначено основні вимоги для створення майбутнього проекту. Спроектовано програмний модуль згідно вимог до продукту. Вибрано набір інструментів для розробки за допомогою, яких розроблений вихідний програмний модуль.

Ключові слова: дослідження, аналіз, фільтрація, спектр, візуалізація, проектування, розробка, Qt. QML.

Abstract

The basic requirements for making future project. Designed software module according to product requirements. Your set of development tools for help, which developed the original software module.

Keywords: research, analysis, filtering, spectrum, visualization, design, development, Qt. QML.

Вступ

Широке впровадження сучасних комп'ютерних технологій і застосування пакетів прикладних програм докорінно змінило і значно полегшило процес оброблення й аналізу медичних даних. При цьому для застосування основних статистичних методів оброблення медичних даних лікарю не потрібно заглиблюватися в складність математичних процедур, але варто зрозуміти, для чого і як ці методи використовуються, а також вміло використовувати обраний пакет прикладних програм.

Результати дослідження

Процес проектування складного програмного забезпечення починається з уточнення його структури, тобто визначення структурних компонентів і зв'язків між ними. Тому слід спершу розглянути сучасні принципи проектування ПЗ, а також спроектувати програмне забезпечення для візуалізації біомедичних сигналів, як на фізичному так і логічному рівні. Розробка продукту передбачає створення зручного багатофункціонального комплексного додатку для аналізу вхідного багатоканального сигналу, який можна було використовувати в не лише в медичних цілях. Програма повинна мати велику кількість вбудованих інструментів для візуалізації вхідного сигналу, його модифікації, а саме головне порівняння та аналізу вихідних даних. Крім того має бути можливість розширення функціоналу по мірі потреб, що потребувало написання добре підтримуваного коду.

Отже основним користувачем звісно виступає особа, яка саме і бере участь у процесі аналізу вхідного сигналу. Вона має отримати доступ до великої кількості інструментів, які мають бути представленні зручним користувацьким інтерфейсом. Крім того кожен інструмент повинен бути не лише простий у використанні, а й максимально конфігуруватися як користувачем під час роботи так і програмістом з подальшим розвитком продукту [1].

Основними елементами є інструменти для управління сценою, які в свою чергу можна поділити на інструменти для:

1. створення нових представлень;
2. редагування та видалення уже створених;
3. керування часовим інтервалом, що досліджується.

В то самий час окрім особи яка проводить аналіз, програма може використовуватися програмістами для відлагодження, за допомогою логування. Згідно проаналізованого матеріалу було побудовано, UML діаграму варіантів використання, яка наведена на рисунку 1

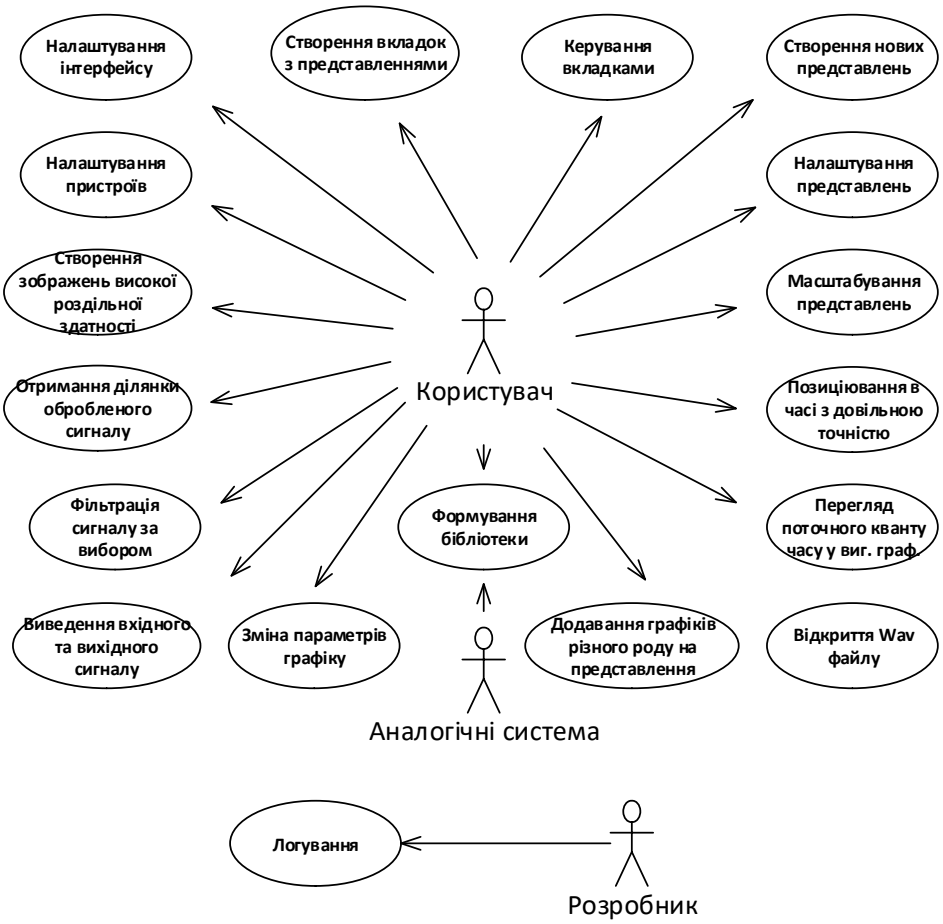

Рисунок 1 – Діаграма варіантів використання.

Особливістю програмного забезпечення, що розробляється, є його функціональна завантаженість. Така форма передбачає використання великої кількості інструментів, а також складних елементів користувацького інтерфейсу, що утворюють окремі модулі із власними залежностями один від одного. Зв'язок між модулями має бути тісний, але частина з них маж бути більш незалежна для того щоб можна було додавати новий функціонал, без серйозних змін в інших компонентах [2].

Оскільки програма має працювати з біометричними сигналами представленні у багатопотокову аудіо файлі, то варто використовувати бібліотеку або ряд бібліотек для стандартизованого аналізу аудіо потоку, а також для отримання спектру. Для уникнення проблем пов’язаних з типізацією даних та використання з іншими бібліотека, слід створити обгортку на вибрану бібліотеку.

Для виводу графіки на екран варто використовувати бібліотеку яка б дозволила використовувати апаратні ресурси машини для виводу графіки (особливо відеоадаптер). Це дозволить розвантажити центральний процесор і дасть розширити функціонал програми без втрати швидкодії. Як із попередньою бібліотеко слід обгорнути бібліотеку, а також розробити додаткові модулі для виводу складних графічних елементів, приведених до об’єктів [1].

Оскільки програма матиме досить складний і багатий візуальний інтерфейс слід вибрати framework (набір бібліотек), який не лише дозволив використовувати стандартні елементи користувацького інтерфейсу (user interface UI), а й спростити роботу з мовою програмування і дати можливість створювати власні елементи UI за допомогою одної із багатьох мов розмітки та задання стилю.

Крім вище сказано потрібно щоб вибраний framework мав можливість роботи з різними типами даних. Не менш важливим моментом є можливість переносу коду на інші платформи і максимальна інтегрованість об’єктних принципів програмування.

Згідно проаналізованого матеріалу та технічного завдання було побудовано діаграму компонентів зображену на рисунку 2.

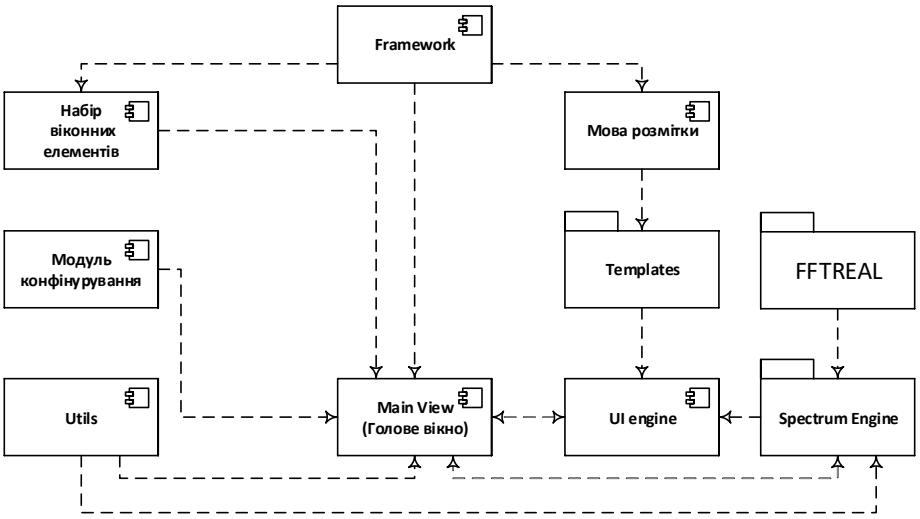

Рисунок 2 – Діаграма компонентів.

Головне вікно має представлятися як ключовий елемент користувацького інтерфейсу. Він має давати доступ до більшості інструментів програми. Сам елемент виконує роль контейнера для візуального інтерфейсу, що має бути підготовленим UI Engine. Такий підхід дозволить позбавитися зайвих залежностей і створювати незалежні модальні вікна із простим користувацьким інтерфейсом, що не лише спростить розробку, а й прискорить її.

Інтерфейс користувача майбутнього додатку, є не стандартним і тому є потреба в розробці власних елементів користувацького інтерфейсу, за допомогою специфічної мови розмітки, які мають знімати обмеження щодо відображення та керування. Не менш важливим є використання апаратного прискорення для відображення усіх складових інтерфейсу.

Для цього потрібно сотворити або розробити власний модуль відображення (UI Engine) який відповідав усім вимогам наведених вище. Крім того необхідна повна підтримка динамічної зміни абсолютно усіх елементів сцени, а також можливість відділити програмний код користувацького інтерфейсу від коду обробки даних. Такий модуль повинен мати змогу працювати з так званими шаблонами (Templates), це дозволить уникнути повторювання коду, а також надасть можливість уніфікувати зовнішній вигляд майбутнього продукту.

Модуль, що виконуватиме зчитування та обробку вхідного потоку даних, його відтворення на вибраний аудіо пристрій, а також підготовлюватиме дані для візуалізації, діаграма зображена на рисунку 3. Модуль має містити компоненти які б дозволили йому, опрацьовувати вхідний багатоканальний wav файл, зчитуючи формат даних із заголовку файлу, а також створювати вихідні файли після обробки.

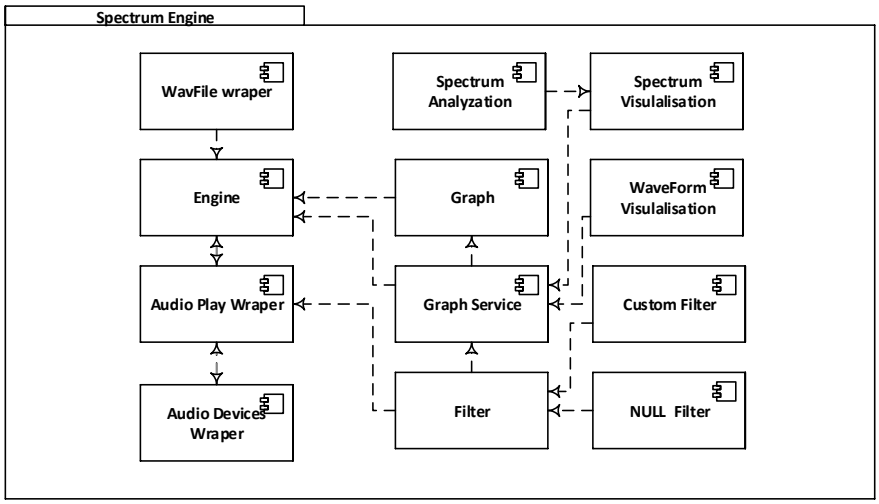

Рисунок 3 – Діаграма Spectrum Engine

Модуль має підготовлювати дані для побудови графіків спектру та форми сигналу, для цього потрібно розробити під-модулі WaveForm Visulalisation та Spectrum Visulalisation. Так як побудувати спектр в певному діапазоні та з певною точністю, не легка задача Spectrum Visulalisation тісно пов'язаний з модулем Spectrum Analyzation, який в свою чергу використовує бібліотеку для швидкого перетворення Фур'є.

Оскільки програма повинна обробляти та відображати декілька потоків одночасно, то підготовка даних має відбуватися в окремих потоках, а також мати зручний сервіс для створення нових об'єктів, які б містили всі необхідні дані для відображення, в ролі такого сервісу виступає Graph Service. Даний сервіс дозволить конфігурувати набір фільтрів які будуть накладатися на вхідний потік даних. Таким чином можна буде створити декілька графіків, представлені об'єктами Graph, які б використовували одні й ті ж дані, а не дублювали їх.

Найбільш привабливим рішенням для розробки згідно вище сказаного є використання бібліотеки класів Qt оскільки вона дає використовувати всі переваги мови C++, яка є найшвидшою хоча і не самою зручною із сучасних мов програмування. Вибрана платформа є не новою, тому її надійність значно спрощує розробку програмного продукту, а інструмент Qt Quick та мова QML надають можливість створити складний користувацький інтерфейс, як незалежну одиницю [3].

Не менш важливим критерієм є те що бібліотека підтримує, велику кількість платформ для розгортання з максимальною адаптивністю до звичних компонентів, а також безкоштовне ліцензування більшості компонентів для корпоративних рішень.

Головне вікно побудоване на основі класу QMainWindow. Даний клас дозволяє отримати доступ до стандартних елементів користувацького інтерфейсу, ініціалізація даного класу наведена нище:

```
MainWidget::MainWidget(QWidget *parent):QMainWindow(parent), m_mode(NoMode),
m_engine(new Engine(this)), m_progressBar(new ProgressBar(this)), m_spectrograph(new
Spectrograph(this)), m_levelMeter(new LevelMeter(this)), ui(new Ui::MainWindow),
m_settingsDialog(new SettingsDialog(m_engine->availableAudioInputDevices(),
    m_engine->availableAudioOutputDevices(), this)){
      m_spectrograph->setParams(SpectrumNumBands, SpectrumLowFreq, SpectrumHighFreq);
      createUi();   connectUi();
}
```

Основною одиницею в UI є QML елементи, розташування яких наведена на рисунку 4.

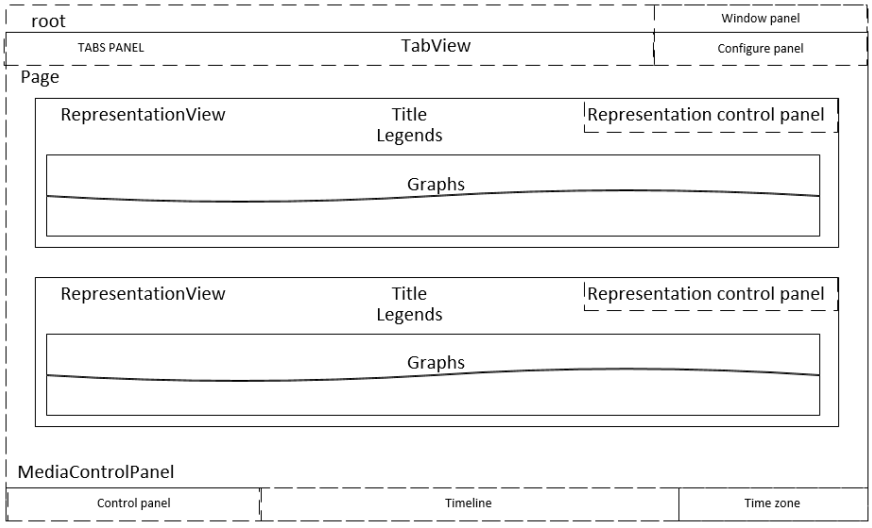

Рисунок 4 – Розташування QML елементів на сцені

Розглянемо кожний елемент окремо:

1. Root – головний контейнер, який містить усі елементи UI, а також ряд констант для їх налаштувань;

2. Window panel – містить набір кнопок та функцій для переходу у повноекранний режим роботи та особливостей роботи в ньому;

3. TabView – контейнер який містить усі створені робочі вкладки, а також їх контент;

а. Tabs Panel – містить набір кнопок які дозволяють перейти на створену вкладку, видалити цільову вкладку або створити нову;

б. Page – являє собою контент вибраної вкладки;

в. RepresentationView – є контейнером для представлення, який місить заголовок та легенду поточного представлення;

г. Representation control panel – дає можливість видалити поточне представлення, а також зконфігурувати його.

4. Configure panel – містить набір кнопок, що дають доступ до конфігурування пристроїв виводу, вікон спектрального аналізу та можливість створювати нові представлення у поточній вкладці;

5. Media Control Panel – є контейнером для панелей керуванням медіа контентом.

а. Control panel – містить кнопки та функції для відкриття файлу для опрацювання і управлінням процесом відтворення;

б. Timeline – відображає поточний час відтворення у процентному співвідношенні, а також дозволяє візуально змінити поточний час відтворення;

в. Time zone – являє собою цифровим варіантом Timeline.

Кожен елемент є автономний та написаний з нуля, що дозволяє досягнути бажаного функціоналу та поведінки створеного елементу. Весь інтерфейс спроектовано в строгому та популярному дизайні у темних тонах для максимальної зручності в роботі, а використання елементів анімації дозволи зробити роботу з програмою більш плавною та зрозумілою. Для підготовки до друку інтерфейс переводиться у світлі тони, а роздільна здатність вихідного зображення зростає до 600dpi. Узагальнений приклад такого елементу, а саме панелі відображення поточного часу відтворення та його зміни, наведено нижче:

```
Row{
   TextInput{
      id: hours
   }   Text {
      id: hoursTitle
   }   TextInput{
      id: minutes
   }   Text {
      id: minutesTitle
   }   TextInput{
      id: seconds
   }
}
```

Хоча кожен елемент є автономною одиницею, він все одно реагує на зміні у середовищі за вдяки своєму батьку, наприклад вище згадана панель реагує на зміну поточного часу за допомогою “Timeline”, у якого з ним спільний батько. Зовнішній вигляд елементів наведено на рисунку 5.

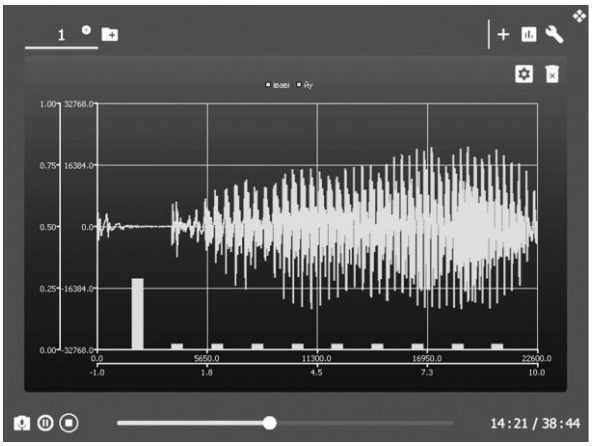

Рисунок 5 – Зовнішній вигляд підготовленої сцени

Крім видимих елементів UI, є безліч спливаючих вікон, які дозволяють конфігурувати фільтри та графіки. Організація обміну даних між C++ та QML здійснюється інтегрованими засобами Qt.

Клас організований на основі основного класу QObject. Наслідування дає можливість використовувати внутрішню систему обміну повідомленнями, схожу на ту що використовується для обміну повідомленнями між елементами UI. Клас містить поля та функції для доступу до пристроїв вводу/виводу звуку, а також для управління потоком даних (старт/зупинка/пауза відтворення, зміна позиції відтворення):

```
const QList<QAudioDeviceInfo> &availableAudioOutputDevices() const
                    { return m_availableAudioOutputDevices; }
bool loadFile(const QString &fileName);
void startRecording();
void startPlayback();
void suspend();
void stopPlayback();
void setTimePosition(qint64 time);
```

Увесь час представлений у вигляді мікросекунд, що дає можливість точно задати проміжок часу для відтворення, для зберігання використовується 64 біт. Даний клас виступає контролером для решти класів. Він повідомляє решту класу про зміну формату вхідних даних, позиції відтворення, зміну статусу відтворення і т.д., але не здійснює безпосередню обробку сигналу. Повідомлення відсилаються за допомогою так званих сигналів (signals), головні з яких наведені нижче:

```
signals:
void stateChanged(QAudio::Mode mode, QAudio::State state);
void infoMessage(const QString &message, int durationMs);
void errorMessage(const QString &heading, const QString &detail);
void formatChanged(const QAudioFormat &format);
void playPositionChanged(qint64 position);
void bufferChanged(qint64 position, qint64 length, const QByteArray &buffer);
```

Зчитування даних відбувається по таймеру у два потоки, що забезпечує отримання даних як для візуалізації так і для звукового супроводу. Після того як чергова порція даних зчиталися, про це повідомляється за допомого сигналу bufferChanged. Для зчитування даних з файлу потрібно отримати інформацію про канали даних, що містяться у файлі який аналізується. Для отримання даних про потоки потрібно прочитати заголовок файлу, в якому присутня інформація про розмір заголовка, кількість каналів, частота дискретизації, кількість біт на «семпл», розташування старшого біту і т.д.. Такий функціонал реалізовує клас WavFile наслідуваний від класу QFile.

Клас містить інформацію про доступні фільтри, а також зберігає і дозволяє створювати нові GraphFilterService, які здійснюють обробку потоку даних отриманих від Spectrum Engine. GraphFilterService являє собою набір фільтрів, а також модулів для підготовки вихідних даних для відображення графіків на представленнях. Кожен сервіс може формувати дані, для декількох представлень. Робота даного сервісу зображено на рисунку 6.

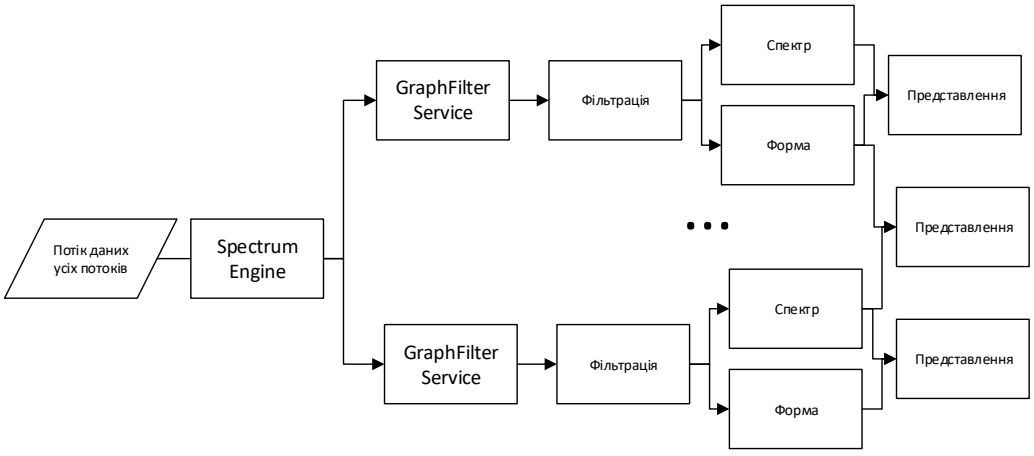

Рисунок 6 – Алгоритм роботи GraphFilterService

Для того щоб, графік представлення мав доступ до даних сервісу, його слід підписати на розсилку нових значень, за допомогою методів subscribeWaveForm та subscribeSpectrum. Дані методи приймають адресу масиву з точками певного графіку, а також номер каналу з якого слід отримувати дані. Після виконання підписки сервіс автоматично обновлює стан графіків в залежності від типу графіку. Якщо графік видаляється слід виконати обернену операцію до підписки за допомогою методів unsubscribeWaveForm та unsubscribeSpectrum.

Фільтрація виконується за допомогою методу doFilter класу що реалізує інтерфейс Filter. Існує два види фільтрів, це NullFilter та CustomFilter. Перший не виконує ніяких операцій на даними і дає можливість отримати прозорі дані. Другий є фільтром 128 порядку параметром якого є вхідна комбінація 128 параметрів фільтру. Підхід з прозорим фільтром дозволяє використовувати єдиний механізм для відтворення як фільтрованих так і не фільтрованих даних. Для отримання даних спектру використовується три класи SpectrumAnalyser, SpectrumAnalyserThread та Spectrograph.

SpectrumAnalyser, SpectrumAnalyserThread класи які відповідають за обрахунок спектру в окремому потоці для кожного набору фільтрів. Основними методами є calculate, calculateSpectrum відповідно. Метод calculateSpectrum приймає масив семплів, а також вхідну частоту, та розмір семпла. Алгорим роботи методу наведений на рисунку 7 [4].

Після отримання значень амплітуд та спектру, вони передаються об’єктам класів WaveFormCustom та Spectrograph. Класи опрацьовують вхідний потік даних та підготовлюють їх до виводу на екран. Обидва класи працюють з даними у форматі QXYSeries. Даний формат представляє собою координати X та Y. Код підготовки даних спектрального аналізу наведено нижче:

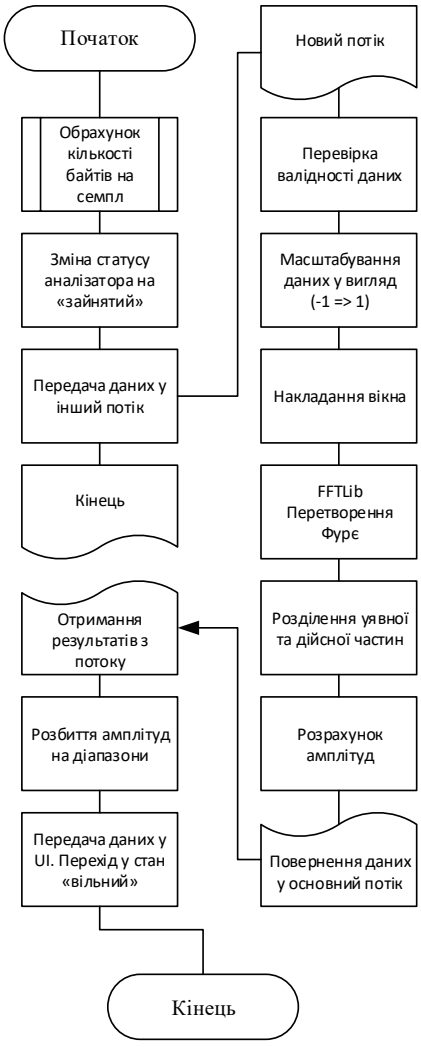

Рисунок 7 – Алгоритм обрахунку та підготовки даних спектру

```
    FrequencySpectrum::const_iterator i = m_spectrum.begin();
    const FrequencySpectrum::const_iterator end = m_spectrum.end();
    const int numBars = m_bars.count();
    for(int j = 0; j < subscribtionList.size(); j++)  {
       subscribtionList[j]->clear();
       for (int i=0; i<numBars; ++i) {
          for (int var = 0; var <= qFloor(m_bars[i].value*50); var++)
             subscribtionList[j]->append(QPointF(i, (float)var/50));
          subscribtionList[j]->append(QPointF(i, m_bars[i].value));
}
```

Обновлення даних UI реалізовано на основі так званих підписок. Будь яке представлення має змогу підписатися на отримання даних з того чи іншого GraphFilterService, при чому або спектру, або форми сигналу, або одночасно обох. Такий підхід дає змогу з економити ресурси ЦП, оскільки підготовка даних здійснюється однин раз, але отримувати дані можуть зразу декілька графіків.

Висновки

Розглянуто особливості проектування програмного забезпечення з використанням об'єктно орієнтованого підходу за допомогою універсальної мови проектування UML. Побудовано діаграму використання, що відображає можливості користувача, а також ряд діаграм для визначення необхідного набору вимог та інструментів для розробки програмного продукту.

Вибрано мову програмування та набір бібліотек, які полегшили написання програмного продукту. Розроблено різноманітні модулі програми для, відображення та формування аудіо контенту, створення та відображення фільтрів у вигляді спектру та форми сигналу. Розроблено програмну реалізацію модуля візуалізації результатів фільтрації та спектрального аналізу звукових біометричних сигналів із зручним і зрозумілим інтерфейсом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. – 2-е изд. – СПб.: Питер, 2006. — 736 с. — ISBN 5-46900-599-2.
2. Крэг Ларман Применение UML 2.0 и шаблонов проектирования – 3-е изд. / Крэг Ларман – М.: Вильямс, 2006. — 736 с. — ISBN 0-13-148906-2.
3. Макс Шлее Qt 5.3 Профессиональное программирование на C++. / Шлее Макс – СПб.: «БХВ-Петербург», 2015. – с. 928. – ISBN 978-5-9775-3346-1.
4. Qt Charts : [Електронний ресурс] – режим доступу до ресурсу : http://doc.qt.io/qt-5/qml-qtcharts-chartview.html

Гайдучок Микола Анатолійович — студент групи 2КІ-16м, факультет інформаційних технологій та комп'ютерної інженерії, Вінни-ця, e-mail: nico13051995@gmail.com.

Науковий керівник: ***Черняк Олександр Іванович*** — канд. техн. наук, доцент кафедри обчислювальної техніки, Вінницький національний технічний університет, м. Вінниця.

Haiduchok Mykola A. — student of the group 2KI-16m, faculty of information technologies and computer engineering, Vinnytsia, e-mail: nico13051995@gmail.com.

Supervisor: ***Chernyak Alexander Ivanovich*** — Cand. Sc.(Eng), Associate Professor of Computer Science, Vinnytsia National Technical University, Vinnytsia